E 43

43.1 EMMS—Empty MMX™ State

Opcode	Instruction	Description
0F 77	EMMS	Set the FP tag word to empty.

Description

Sets the values of all the tags in the FPU tag word to empty (all ones). This operation marks the MMX registers as available, so they can subsequently be used by floating-point instructions. (See "FPU Tag Word", for the format of the FPU tag word.) All other MMX instructions (other than the EMMS instruction) set all the tags in FPU tag word to valid (all zeros).

The EMMS instruction must be used to clear the MMX state at the end of all MMX routines and before calling other procedures or subroutines that may execute floating-point instructions. If a floating-point instruction loads one of the registers in the FPU register stack before the FPU tag word has been reset by the EMMS instruction, a floating-point stack overflow can occur that will result in a floating-point exception or incorrect result.

Operation

```
FPUTagWord ← FFFFH;
```

Flags Affected

None.

Protected Mode Exceptions

#UD	If EM in CR0 is set.
#NM	If TS in CR0 is set.
#MF	If there is a pending FPU exception.

Real-Address Mode Exceptions

#UD	If EM in CR0 is set.
#NM	If TS in CR0 is set.
#MF	If there is a pending FPU exception.

Virtual-8086 Mode Exceptions

#UD	If EM in CR0 is set.
#NM	If TS in CR0 is set.
#MF	If there is a pending FPU exception.

43.2 ENTER—Make Stack Frame for Procedure Parameters

Opcode	Instruction	Description
C8 *iw* 00	ENTER *imm16*,0	Create a stack frame for a procedure
C8 *iw* 01	ENTER *imm16*,1	Create a nested stack frame for a procedure
C8 *iw* ib	ENTER *imm16,imm8*	Create a nested stack frame for a procedure

Description

Creates a stack frame for a procedure. The first operand (size operand) specifies the size of the stack frame (that is, the number of bytes of dynamic storage allocated on the stack for the procedure). The second operand (nesting level operand) gives the lexical nesting level (0 to 31) of the procedure. The nesting level determines the number of stack frame pointers that are copied into the "display area" of the new stack frame from the preceding frame. Both of these operands are immediate values.

The stack-size attribute determines whether the BP (16 bits) or EBP (32 bits) register specifies the current frame pointer and whether SP (16 bits) or ESP (32 bits) specifies the stack pointer.

The ENTER and companion LEAVE instructions are provided to support block structured languages. The ENTER instruction (when used) is typically the first instruction in a procedure and is used to set up a new stack frame for a procedure. The LEAVE instruction is then used at the end of the procedure (just before the RET instruction) to release the stack frame.

If the nesting level is 0, the processor pushes the frame pointer from the EBP register onto the stack, copies the current stack pointer from the ESP register into the EBP register, and loads the ESP register with the current stack-pointer value minus the value in the size operand. For nesting levels of 1 or greater, the processor pushes additional frame pointers on the stack before adjusting the stack pointer. These additional frame pointers provide the called procedure with access points to other nested frames on the stack. See "Procedure Calls for Block-Structured Languages", for more information about the actions of the ENTER instruction.

Operation

```
NestingLevel ← NestingLevel MOD 32
IF StackSize = 32
    THEN
        Push(EBP) ;
        FrameTemp ← ESP;
    ELSE (* StackSize = 16*)
        Push(BP);
        FrameTemp ← SP;
FI;
IF NestingLevel = 0
    THEN GOTO CONTINUE;
FI;
IF (NestingLevel > 0)
    FOR i ← 1 TO (NestingLevel − 1)
        DO
            IF OperandSize = 32
                THEN
                    IF StackSize = 32
                        EBP ← EBP − 4;
                        Push([EBP]); (* doubleword push *)
                    ELSE (* StackSize = 16*)
                        BP ← BP − 4;
                        Push([BP]); (* doubleword push *)
                    FI;
                ELSE (* OperandSize = 16 *)
                    IF StackSize = 32
                        THEN
```

```
                              EBP ← EBP − 2;
                              Push([EBP]); (* word push *)
                          ELSE (* StackSize = 16*)
                              BP ← BP − 2;
                              Push([BP]); (* word push *)
                      FI;
              FI;
    OD;
    IF OperandSize = 32
        THEN
            Push(FrameTemp); (* doubleword push *)
        ELSE (* OperandSize = 16 *)
            Push(FrameTemp); (* word push *)
    FI;
    GOTO CONTINUE;
FI;
CONTINUE:
IF StackSize = 32
    THEN
        EBP ← FrameTemp
        ESP ← EBP − Size;
    ELSE (* StackSize = 16*)
        BP ← FrameTemp
        SP ← BP − Size;
FI;
END;
```

Flags Affected

None.

Protected Mode Exceptions

#SS(0)	If the new value of the SP or ESP register is outside the stack segment limit.
#PF(fault-code)	If a page fault occurs.

Real-Address Mode Exceptions

#SS(0)	If the new value of the SP or ESP register is outside the stack segment limit.

Virtual-8086 Mode Exceptions

#SS(0)	If the new value of the SP or ESP register is outside the stack segment limit.
#PF(fault-code)	If a page fault occurs.